TD 3E204

2020-2021

TD1 : GPIO

EXERCICE 1 – CODAGE

Coder en base 2 les nombres décimaux suivants :

- 12
- 1024
- 345

Coder en base 16 les nombres binaires suivants :

- 100101
- 11110000
- 10101010

Coder en base 16 les nombres décimaux suivants :

- 10
- 16
- 458

Coder en complément à deux sur 8 bits les nombres suivants :

- 34
- -23
- -128
- 127

EXERCICE 2 – TYPES DE DONNEES ET MANIPULATION BOOLEENNE EN LANGUAGE C

Dans cet exercice, le terme instruction est compris au sens large et peux signifier plusieurs mots de commande et affectations en langage C. Remarque : `Condition` et `donnée` sont codées sur 8 bits.

1. Quels sont les opérateurs booléens du langage C ?

2. Quels sont les opérateurs sur les champs de bits du langage C ?

3. Ecrire l'instruction qui impose la valeur 1 au 6^{ième} bit de la variable `donnée` sans modifier les autres bits ;

4. Ecrire l'instruction qui impose la valeur 0 au 4^{ième} bit de la variable `donnée` sans modifier les autres bits ;

5. Ecrire l'instruction qui complémente la valeur du 7^{ième} bit de la variable `donnée` sans modifier les autres bits ;

6. Ecrire l'instruction qui impose la valeur 0 à la variable `donnée` si la variable `condition` vaut 0;

7. Ecrire l'instruction qui impose la valeur 15 à la variable `donnée` si la variable `condition` vaut 1 ;

8. Ecrire l'instruction qui impose la valeur 255 à la variable `donnée` si le 3^{ième} bit de la variable `condition` vaut 1 ;

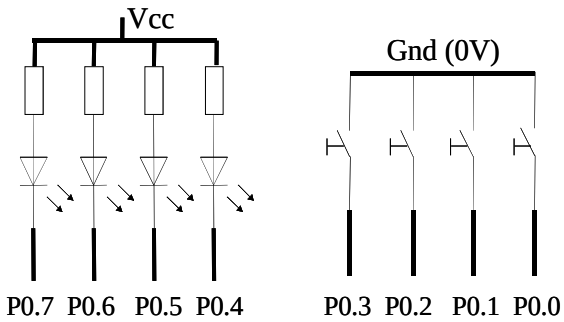
9. Ecrire l'instruction qui impose la valeur 0 à la variable `donnée` si le 5^{ième} bit de la variable `condition` vaut 0 ;

10. Ecrire l'instruction qui impose la valeur 255 à la variable `donnée` si le 4ième bit de la variable `donnée` vaut 1 ou si la variable `condition` vaut 0 ;

EXERCICE 3 - GESTION DU PORT D'ENTREE SORTIE DU MICROCONTROLEUR

On souhaite commander à l'aide de quatre interrupteurs quatre LEDs connectées au port parallèle du microcontrôleur LPC2378 de la façon suivante :

Chaque interrupteur commande une LED (l'interrupteur connecté à P0.0 commande la LED reliée à P0.4 et ainsi de suite…)



Lignes d'E/S du microcontrôleur

1) Comment doit-on configurer le port d'entrée/sortie du microcontrôleur pour réaliser notre exemple ? Doit-on le modifier pour cela ?
2) Quels registres permettent de lire ou d'écrire sur le port parallèle d'entrée/sortie ? Comment doit-on le modifier
3) Ecrire l'instruction permettant d'allumer les LEDs reliées à P0.4 et P0.5 (sans tenir compte des interrupteurs).

4) Ecrire un programme en C commandant l'allumage des LEDs par l'action des interrupteurs.

TD 2
Scrutation et Interruption externe

On considère un montage ou le LPC2378 est connecté avec 8 LEDs sur la partie basse du port P4 (P4.0 à P4.7), 8 interrupteurs sur le deuxième octet du port P4 (P4.8 à P4.15) ainsi qu'un bouton poussoir sur le pin P2.13 du port P2. Ecrire la fonction `init_GPIO()` qui permet de configurez les registres de sélection et de direction du LPC2378 pour respecter le montage considéré ci-dessus.

Partie 1 : Scrutation

On désire à présent faire clignoter les LEDs que lorsque le bouton poussoir est appuyé. Pour cela il faut scruter/tester le bouton poussoir en continu afin de connaitre son état à chaque instant (on parle alors de « polling »).

1. Ecrire un programme afin que le clignotement soit conditionné par l'appui du bouton poussoir (P2.13).

2. Modifiez votre programme afin que les LEDs s'allument lorsque le bouton poussoir est appuyé et s'éteigne sinon.

3. Ecrire un programme qui permet d'allumer les LEDs comme un chenillard :

- Décalage de droite à gauche si on appuie sur le bouton poussoir.
- Décalage de gauche à droite si on n'appuie pas.

Partie 2 : Interruptions externes

1. Ecrire un premier programme simple qui permet d'allumer ou d'éteindre les LEDs à chaque appui du bouton poussoir. Gérer le bouton poussoir par interruption et non plus par scrutation. N'oubliez pas de modifier la fonction init_GPIO() pour configurer la bonne fonction de la PIN P2.13.

2. Modifier votre programme afin qu'après chaque appui du bouton poussoir, les LEDs se mettent à clignoter ou s'arrête de clignoter et reste dans le même état jusqu'au prochain appui sur le bouton poussoir.

3. Ecrire un nouveau programme qui permet de lire la valeur qu'il y a sur les interrupteurs et la recopie sur les LEDs à chaque nouvel appui sur le bouton poussoir.

4. Ecrire un programme qui à chaque appui sur le bouton poussoir génère un compteur qui compte jusqu'a 150 sur les LEDs et s'arrête ensuite.

5. Ecrire un programme qui permet d'allumer les LEDs comme un chenillard, décalage de droite à gauche et ensuite de gauche à droite. Mais le défilement doit être conditionné par le bouton poussoir. A chaque appui sur le bouton poussoir, les LEDs se décalent d'un cran.

TD 3 : Timers

Génération de fréquence pour télémètre à ultrason

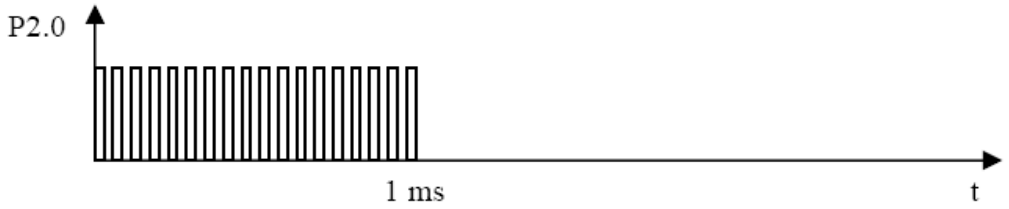
On désire généré un signal qui va venir en entrée d'un télémètre à ultrason afin d'effectuer une mesure de distance. Pour cela, il faut envoyer un signal périodique de 40 Khz pendant 1 ms pour chaque nouvelle mesure. La mesure de distance s'effectue en mesurant le temps du trajet aller/retour de cette onde entre l'émetteur/récepteur du télémètre et l'obstacle.

1. On veut générer sur le bit 0 du port `P2 (P2.0)` un signal périodique avec un rapport cyclique de 50 % et une fréquence de 40 kHz. Pour cela on utilise le Timer 0 du LPC2378 . Rappel : PCLK=14,4 Mhz. Ecrire les fonctions suivantes :

```
init_GPIO() : procédure d'initialisation du port P2
init_T0() : procédure d'initialisation du Timer 0
isr_T0() : sous-programme d'interruption du Timer 0
main() : programme principal
Vérifier le résultat avec le simulateur en utilisant le "Logic Analyser".
```

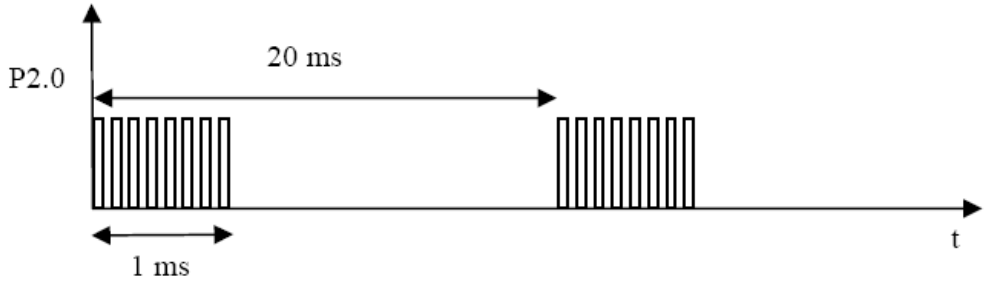
2. On ne veut émettre ce signal à 40 kHz que pendant 1 ms. Pour cela on utilise le timer T1 pour arrêter T0 et T1 au bout de 1 ms. Écrire ou modifier les procédures suivantes.

```
init_T1() : procédure d'initialisation du Timer T1
isr_T1() : sous-programme d'interruption du Timer T1
main() : programme principal
Vérifier le résultat avec le simulateur en utilisant le "Logic Analyser".
```

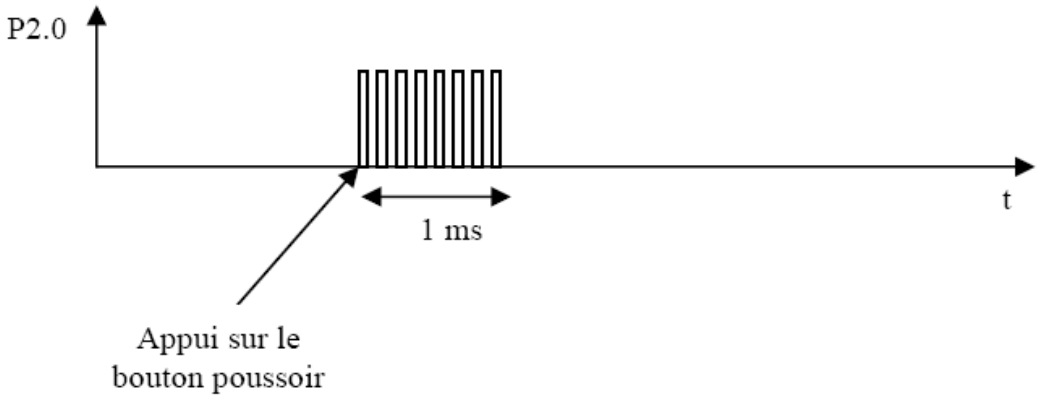


3. On veut répéter cette impulsion du signal à 50 kHz toutes les 20 ms. Pour cela on utilise le timer 2 pour démarrer les timers `T0` et `T1` toutes les 20 ms. Écrire ou modifier les procédures suivantes :

```
init_T2() : Procédure d'initialisation du timer T2
isr_T2() : Procédure de service d'interruption du timer T2
main() : Programme principal.
Vérifier le résultat avec le simulateur en utilisant le "Logic Analyser".
```



4. Maintenant on veut que le signal de 40KHz soit générer pendant 1 ms à chaque fois qu'un utilisateur appui sur un bouton poussoir qui est connecté sur la broche 10 du port P2 (P2.10). Proposez une solution logicielle pour répondre à cette nouvelle fonctionnalité ?



TD4 : PWM

Dans cet exercice on utilise le périphérique PWM afin de générer des signaux PWM (Pulse Width Modulation) ou MLI en français (Modulation de Largeur d'Impulsion). Le périphérique PWM fonctionne comme un compteur avec quelques fonctionnalités spécifiques. Voir la documentation du LPC2300.

1. Ecrivez la fonction `void init_pwm(void)` et le programme principal `void main(void)` qui permet de configurer les registres du PWM afin de générer un signal avec une période quelconque mais avec un rapport cyclique de 30 %.

2. Modifier votre programme afin de générer un signal avec une période de 40 Khz et un rapport cyclique de 70 % (fréquence Pwm est de 14.4 MHz).

3. Maintenant, on veux faire évoluer le rapport cyclique du signal PWM comme un signal triangulaire. Pour cela, il faut que le rapport cyclique du signal généré évolue comme une rampe. Ecrivez le sous programme d'interruption `void isr_pwm(void)` et rajouter les instructions nécessaires dans la fonction `void init_pwm(void)` afin de respecter ce nouveau cahier des charges.

4. Ecrivez un programme qui permet de configurer le périphérique PWM afin de générer les signaux de la figure suivante (Sample PWM Waveforms).